Discussion 10:
Delayed Expressions and Binary Trees

TA: **Jerry Chen**
Email: **jerry.c@berkeley.edu**
TA Website: **jerryjrchen.com/cs61a**

Agenda

1. Iterators (review)
2. Streams
3. Binary (Search) Trees

Announcements

Scheme proj checkpoints

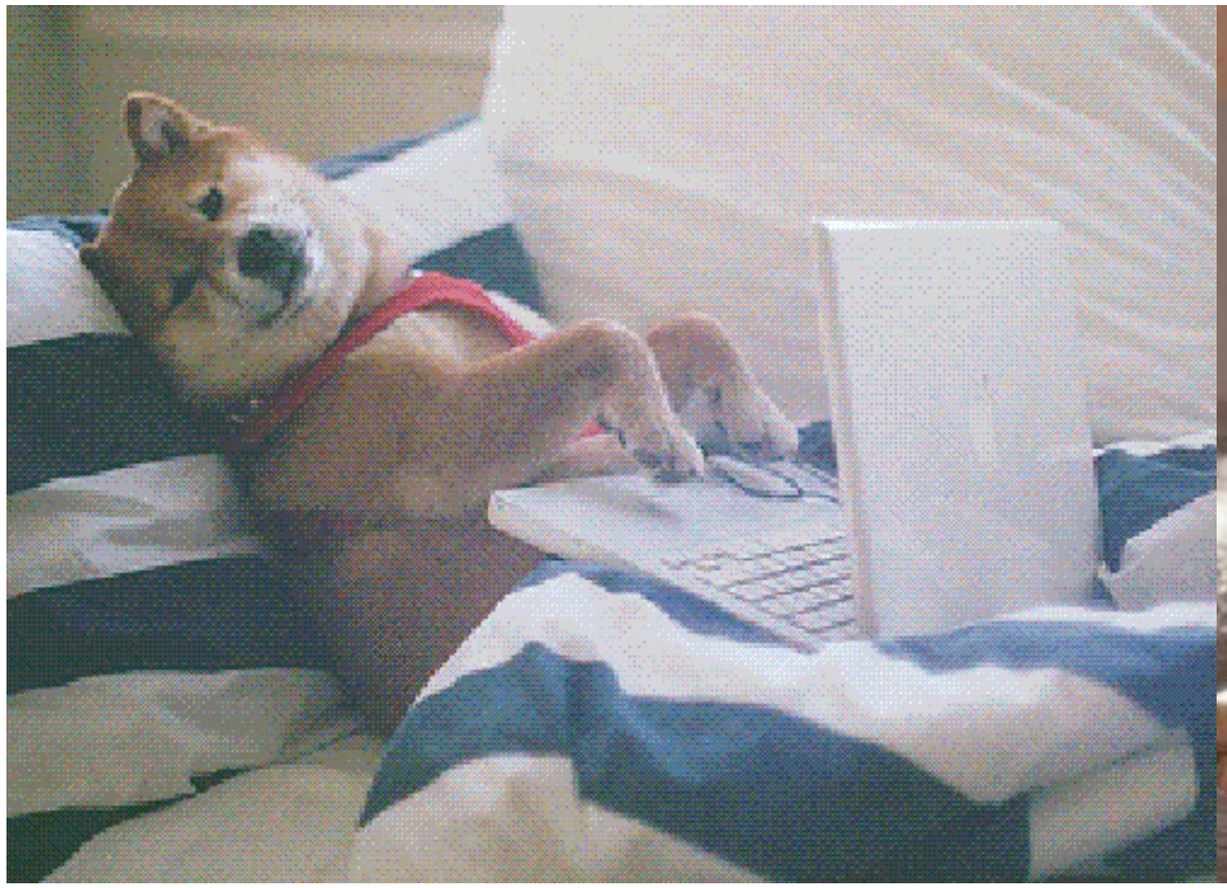
- Part 1 by this Thursday, Part 2 (and proj) by next Thursday
- There are **hidden tests**, through email!

Maps composition due by Sunday 4/16

HW 08 due Tues

Delayed Expressions

"Lazy evaluation"



Iterators/Iterables

Iterable

- **Returns an iterator** using `iter()`

Iterator

- Returns the next **item** in sequences using `next()`
- `next()` (probably) will **modify some state**

Iterators/Iterables

"The iterable is a book, and the iterator is a bookmark"

If something is **iterable**, we can get its **iterator** using `iter()` and examine all its elements by repeatedly calling `next()` on that iterator.

Keep in mind that iterators are usually **one-time use**. Stepping through a sequence again means calling `iter()` again.

Streams

Like a linked list, except **evaluated lazily**

- Don't make rest **until we ask for it**
- After we ask for it, **remember the result**
- **Rules** (functions) tell us how to create the next element

Stream Constructors

Link(<first>, <rest>)

Must be a linked list (incl. empty)

Stream(<first>, <rest>)

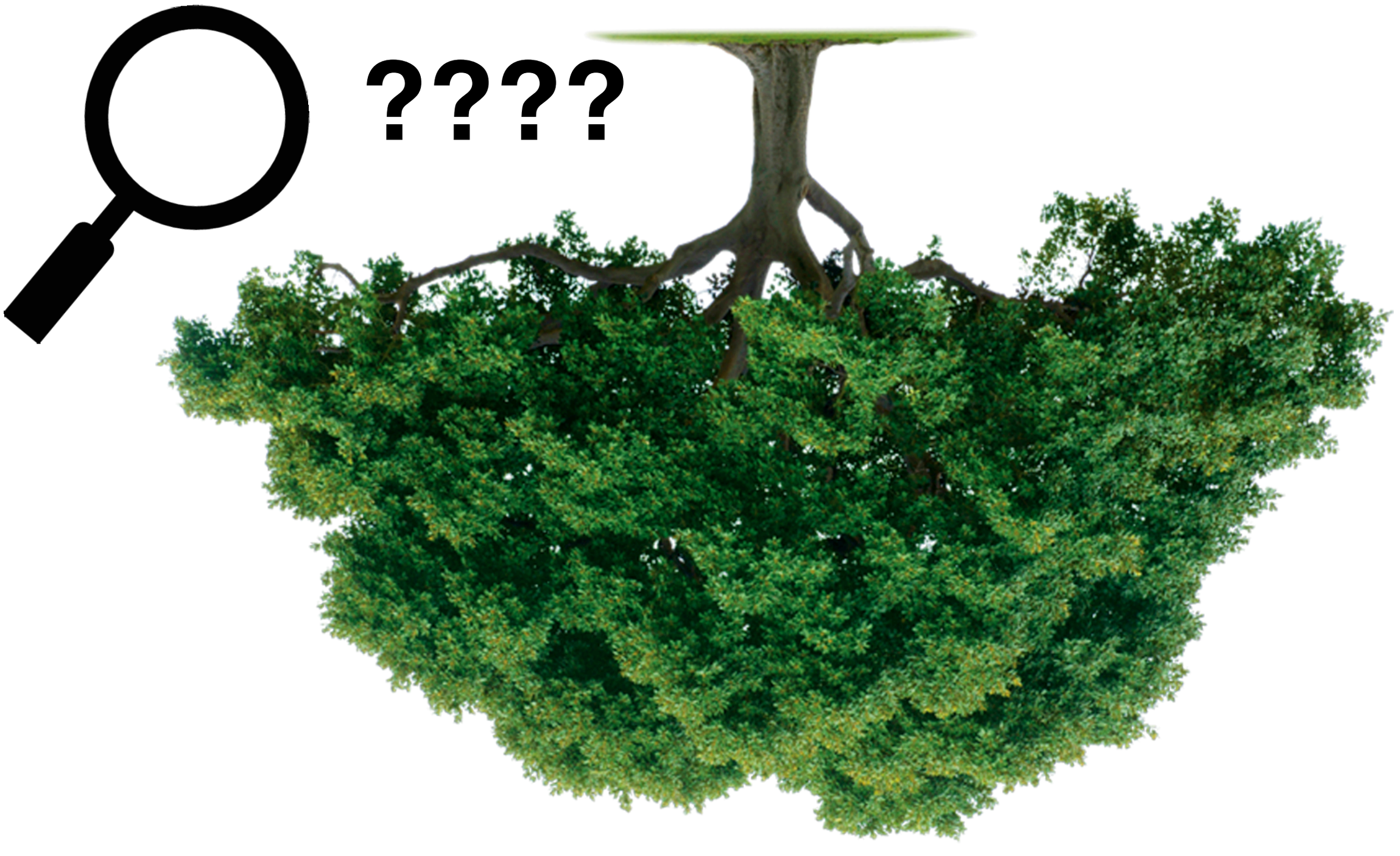
A stream, OR
a function that returns a stream

Streams

Similarities to linked lists:

- `first` gets the **front** of a stream
- `rest` gets the **rest** of a stream
- `Stream.empty` is the **empty** stream

Binary Search Trees



Binary Search Trees

- Often abbreviated as "BST"
- Binary - each tree/subtree has **up to two children**
- Search - efficient, due to our invariant

Binary Search Trees

The BST invariant:

- A BST may have a left and right child
- All values in **left are less than the root**
- All values in **right are greater than the root**